Date: April 3, 2005

From: Shakeel Mustafa
24831 Hendon St
Laguna Hills, CA 92653
Tel: 949-457-1243

To: Ms. Courtney D. Fields (Primary Patent Examiner)
Art Unit 2137
Fax # 1-703-872-9306
US Patent Office

Attention: Ms. Courtney D. Fields

REF: Application/Control Number: **09/848,670**

Dear Ms. Fields:

I am responding to your Detailed Office Action. Attached, you will find my responses to the Objections cited in reference to the above Application. You will also find a list of 23 Amended/New Claims that incorporate and take into consideration the objections raised in the Office Action along with my comments and feedback.

Please contact me via telephone at 949-457-1243 or by mail at my above mailing address. I look forward to working with you. Thanks,

Regards,

Shakeel Mustafa

Shakeel Mustafa

Replacement Sheet

(Illustrating reference characters 60, 69, 70, 71, 84 85 and 90)

[0043] FIGS. 5A, 5B, and 5C demonstrate this concept in more detail. FIG. 5A represents the Function Bits entries for the [Group #1] Group # 0 60. As shown, the location of the bit b_0 61 is measured as x_0 69 number of bits away from the start boundary of the random number `x` 30. Similarly, the location of the next bit b_1 63 is known to the system as x_1 70 number of bits away from the bit b_0 69 position. Further, the system can find the position of the bit b_2 65 as a bit located exactly in the middle of the random number `x` 30 consisting of length L. As illustrated in FIG. 5A, if the said random number consists of even bits, then the position of the bit b_2 65 is located as L/2 71 bits away from the ending boundary. If the said random number contains an odd number of bits then the bit b_2 65 position can be identified as (L+1)/2 71 from the ending boundary.

[0044] Referring back to column 50 in FIG. 4B, the Group Number entry 57 represents the binary Group numeric value of 1 and this number picks b_0, b_1, b_2 ... b_p (p = number of Function Bits) with each of the individual bit positions corresponding to y_0, y_1, y_2, ... y_p from pre-defined boundary lines, respectively. This is shown in FIG. 5B under the Group # 1, 80. It should be noted that each of the individual Group or Function Bits does not have to be located at a different bit locations. Two or more Group or Function Bits can occupy the same bit positions. This is shown in FIG. 5B where the two Function Bits b_0 83 and b_1 81 occupy the same bit locations as identified by y_0 86 or y_1 87. Similarly, Function Bits b_2 84 and b_p 85 can be uniquely identified from the ending boundary of the random number 'x' 30 at the distance of y_2 89 and y_p 82 respectively. FIG. 5C shows the Group # n, 90 which has allocated 'q' number of Functions Bits in the random number 'x' 30. Each of the bit locations is determined by the respective distance vector numbers, e.g., z_0 to z_q, similar to the manner as discussed above.

Replacement Sheet

(Illustrating reference characters 101 and 105 as included in paragraph [0045])

[0045] FIG. 6 maintains the range of all the possible numeric values of the binary numbers (b_k, ... b_2, b_1, b_0) resulting from a given set of Function Bits in a tabular form as shown in column 100. Each possible binary numeric value is associated with a pre-arranged sequence of mathematical or logical functions selected from the function class defined in the first pool 101. As illustrated in FIG. 6, a resulting binary value of 0 indicated by the table entry 103 corresponds to a sequence of mathematical or logical functions (f_1 (x), f_2 (x) ... f_n (x)) 105. Similarly, each resulting numeric value of the Function Bits uniquely corresponds to a single or plurality of the pre-arranged functions.

Replacement Sheet

(Illustrating reference characters 150, 180 and 190 as included in paragraphs [0048] and [0049])

[0048] FIGS. 8A and 8B illustrate an example of a function operation followed by its corresponding inverse function operation on a digital information segment of an arbitrary length. This relationship is illustrated in 150. In the presented example, the operation of the function $g_1(x)$ 155 consists of two operators. The first operator R(m) 153 rotates the bits contained in the information segment 157 towards the right to an equivalent number of `m` bits, 160. In the next step, the second operator 151 adds a binary number `n` 163 to the already rotated information segment resulting in an encrypted information segment 165 consisting of k number of bits. It should be observed that depending upon the type of operations performed on the digital information segment the resulting length of the encrypted information segment could be more or less than the original segment. This difference can be consisted of single or multiple bits. Generally, the digital information is processed and exchanged among the communication layers in term of multiple bytes (8 bits/byte). To ensure that the encrypted information segment is consisted of multiple bytes, a padding header followed by a certain number of padding bits is appended. As shown in FIG. 8A, the padding header 167 consisted of 3 bits indicates that how many padding bits are inserted to make the total encrypted information segment length to be divisible by eight bits or any other number.

[0049] FIG. 8B illustrates the operation of the inverse function of $g_1(x)$, represented as $g^{-1}(x)$ 175, on the encrypted information segment 165. The inverse function $g^{-1}(x)$ 175, by its definition, contains all the necessary operators that can reverse the effects of the operations performed by the function $g_1(x)$. In this sense, the inverse function $g^{-1}(x)$ 175 is consisted of two operators; the first operator 173 represents a subtraction of number `n` while the second operator 171 represents a left rotation equivalent to `m` number of bits. As the encrypted information segment is processed for decryption the first step is to remove the padding header 167 along with the associated bits. Next, the operator 173 subtracts the number `n` 180 from the encrypted information segment 165. As a final step the operator L(m) 171 rotates the said segment towards the left to an equivalent of `m`

bits 190. The resulting information segment 185 is exactly the same as the information segmcnt 157 before the encryption process. It is evident from this example that the contents of any information segment consisted of any arbitrary length remains unchanged first by the operation of the function $g_1(x)$ and then by the operation of its inverse function $g^{-1}(x)$.

Replacement Sheet

(Illustrating reference characters 213, and 225 with proposed amendments for further clarification)

[0054] Referring back to FIG. 9, which illustrates that the remote `B` 5 uses the resulting binary numeric value and refers to the table shown in FIG. 6 to identify the corresponding functions (f_1 (x), f_2 (x) ... f_n (x)) belonging to the first function pool. Once the functions in the first function pool are identified the corresponding functions in the second pool (g_1 (x) ...g_n (x)) can be easily identified from the table given in FIG. 3. As it is evident, a person skilled in the art may choose another way to collaborate the functions in the first pool being used to encrypt random number with the functions in the second pool being used to encrypt data segments. For example, one may use the information contain in the random number or encrypted random number e.g., Group and Function Bits, to choose the functions from the second pool which are used to encrypt data segments. Nevertheless, any of these techniques fall under the scope of the presented invention.
The information segment `D` 205 can either contain system or the user's information. As will be discussed later in the section, the system information is typically used to exchange a modified or updated set of rules associated with Group Bits, Function Bits, mathematical and logical functions, etc. The distinction between user's information and system information is made through the use of a single inserted bit 210. A bit value of 0 represents a user's information segment whereas a bit value of 1 means a system information segment. The plurality of functions identified in the second pool (g_1 (x) ...g_n (x)) operate on the information segment [`S`] 'D' as shown in step 215 and produce the first encrypted information segment Dg 220.

[0055] The plurality of the functions identified from the first function pool (f_1 (x), f_2 (x) ... f_n (x)) operate on the random number `x` 207, modify the characteristics of the said random number and produce another number `y` 217. The modified random number `y` 217 replaces the previous random number `x` 207 and is processed as the new seed random number for the next encryption cycle. It should be noted that the modified random number `y` 217 is not exchanged between the host `A` 1 and the remote `B` 5. As

follows from the preceding discussion, it is necessary that both the said host and the remote must agree in advance about the number of encryption cycles to be performed. This information can be mutually configured in a number of ways. In one embodiment, a pre-determined number of bits located at predetermined bit positions within the modified random number `y` 217, similar to Function or Group Bits, can be mutually reserved. As shown in FIG. 7, the numeric binary value of the pre-selected number of bits (b_e . . . b_2, b_1, b_0) listed in column 117 uniquely corresponds to a number N_T in column 118 which determines the total number of encryption cycles to be performed. Since the number `y` 217 by itself is not exchanged over the communication channel it makes it extremely difficult for a hacker to guess about the number of encryption rounds performed on any given information segment. As it is evident, a person skilled in the art may choose another way to collaborate this information at the both said host and the remote. Nevertheless, any of these techniques falls under the scope of the presented invention.

[0056] Referring back to FIG. 9, the encrypted data segment, Dg 220, is further encrypted by a sequence of "Intermediate Function" as illustrated by block 225. This encryption procedure is repeated for N_T 213 number of times. Each new encryption cycle uses the modified random number produced from the previous routine as a seed random number. The corresponding numeric value of the Group and Function Bits select a new set of unique functions from the first and second function pools. The functions identified in the first function pool encrypt the random number further produced from the previous encryption cycle. The corresponding functions identified in the second function pool further encrypt the information segment. ~~These encryption rounds are repeated as illustrated in step 221~~. As illustrated in the FIG. 9 the last encryption round, N_T 213, produces the seed random number `z` 219. The corresponding numeric value derived from the Group and Function Bits of the said seed number chooses a sequence of functions, $u_1(z)..u_w(z)$ from the first pool. The corresponding functions identified from the second pool operate on the already encrypted information segment D_{gh} 230 in step 235 to produce a further encrypted information segment resulting as D_{ghv} 240. It should be noted that any further encryption process stops after the execution of N_T 213 number of encryption rounds. A padding header with the appropriate number of padding bits is

appended, if necessary, to convert the said encrypted information segment so it is divisible by a desired number. The resulting segment is passed down to the lower communication layers for a reliable delivery to said host `A` 1. The packet 300 carries the encrypted segment D_{ghv} with the appropriate header (H) and trailer (T) appended by the lower communication layers for proper delivery. As pointed out earlier it is the responsibility of the lower communication layers to guarantee the transmission of packet 300 to its final destination host `A` 1.

Replacement Sheet

(Illustrating reference characters 309, 320, and 321 with proposed amendments for further clarification))

[0057] Now referring to the host side of FIG. 9, the host `A` 1 receives the random number `x` contained in packet 200. During the information delivery process the lower level communication layers ensure that the contents of the packet are unchanged. As the said random number is processed it yields the exact same Group Number resulting from the Group Bits as produced by the remote `B` 5. The Function Bits and their corresponding numeric value are also the same as well. Since the resulting binary numeric value is the same, the functions (f_1 (x), f_2 (x) ... f_n (x)) selected from the first pool and the corresponding functions (g_1 (x) ...g_n (x)) identified from the second pool will also be the same as used by remote `B` 5 in its first encryption cycle. Remote `B` 5 identifies the inverse functions corresponding to each of the functions selected from the second pool. The resulting sequence of identified inverse functions (g^{-1} . . . g^{-n}) are tabulated in the opposite order (g^{-n} . . . g^{-1}) 321 in a decrypting function table 325. The selected functions from the first pool (f_1 (x), f_2 (x) ... f_n (x)) operate on the random number `x` 207 to produce the next number `y` 217. It should be noted that the resulting number `y` 217 is exactly the same on the both host `A` 1 and remote `B` 5 sides since it results from the same operations performed on the random number `x` 207.

[0058] As explained earlier with reference to FIG. 7, the host `A` 1 also determines the number N_T 329 exactly through the same procedure as used by the remote `B` 5. The both said host and the remote come up with the same number N_T, since they both use exactly the same bits (b_e . . . b_2 b_1 b_0) value from the number `y` 217 and also consult the identical table. In the next round of the encryption cycle the number `y` 217 is used as a seed random number. The corresponding Group and the Function Bits are identified and then subsequently the sequence of functions from the first and the second pool are identified. The inverse of each of the functions identified in the second pool are appended in the decryption function table 325 in the opposite order. The sequences of inverse functions that are identified through this operation are assembled in the "Inverse

Intermediate Function" block 320. The block 320 contains the inverse functions of all the functions identified and included in the "Intermediate Functions" block 225, but are arranged in the opposite order. The same procedure is repeated for N_T times and at the end of each round the resulting inverse functions are identified and tabulated in the decryption function table 325. Although no information segment has been received yet from remote `B` 5 for decryption, but the inverse functions populated in the decryption function table 325 are ready to perform an exact opposite sequence of operations on any digital information segment that may receive from the remote `B` 5.

[0059] As the part of the last encryption cycle the original seed random number `x` 207 is changed to a number `z` 219 which is also exactly the same number `z` 219 as resulted at remote `B` 5. This implies that host `A` 1 will also select the same sequence of functions $(u_1(z)\ldots u_w(z))$ and the corresponding inverse functions in the opposite order $(v^{-w}\ldots v^{-1})$ 309 which will be tabulated accordingly in the decryption function table 325.

Replacement Sheet

(Illustrating reference characters 430, 435, 505, 510 and 555 with the underlined proposed amendments for further clarification)

[0069] With the use of the extension bit, the said length field can be extended to include multiple bytes. It should be noted that the protocol 400 can contain any number of random bits padded after the data bytes. This provision ensures that the length of the protocol 400 can consist of any arbitrary number of bits that may be more suitable for the encryption process. The location of the first Group Number bit, G_0 420, corresponds to a bit distance of d_0 (defined in number of bits) from a specific boundary line of a random number. For example, the bit field 415 can be used to define a reference boundary line. A bit value of 1 in the said field may be attributed to the starting boundary line located at the right end of the random number, whereas a value of 0 indicates that d.sub.0 is measured in bits from the left end of the random number. The field 425 represents the extended bit field concept. This means that if the value of the said field is 0 then the present d.sub.0 field is extended to include the next 7 bits of the next adjacent byte to represent its value. This simple technique ensures that if the d_0 field needs to represent a large number then it can span to include multiple numbers of the next adjacent bytes. The receiving end accordingly identifies the end of a particular field by finding a value of 1 in the bit field 425. In a similar fashion, all the Group Bits (G_1) 430 to G_n 435 locations are identified by the corresponding d_0, d_1 . . . d_n, values, respectively. The CRC field 440 may be included to provide an extra protection for data integrity, even though it is the responsibility of the transport layer and the layers below to ensure that the data has been properly delivered.

[0070] The next protocol format 450 is used to send the exact location of the binary Function Bits inside a random number exclusively used by a particular Group Number. The field 455 presents the instruction mnemonic reserved for this purpose. The field 460 identifies the particular Group Number for which the corresponding binary Function Bits location in a random number needs to be identified. The length field 410 represents the total data bytes to be followed. As mentioned earlier, any number of padded bits can be

appended after the data bytes to make it more suitable for encryption. In the next field, x_0 represents the distance in bits that can be used to identify the position of the function bit b_0 465 in any random number. Similarly, the following fields from x_1 to x_k represent the bit distance for the corresponding Function Bits b_1 470 to b_k 475, respectively.

[0071] The protocol format 480 provides a way to exchange information about the Function Bits binary numeric value and the associated logical or mathematical function to be performed on an information segment. The field 485 presents a unique instruction for this purpose. The length of field 490 shows the number of data bytes to be followed. The field B1 495 identifies the Function Bits binary numeric value with the associated number value for the function $f_1(x)$ which is included in the next field 500. Similarly, all the Function Bits binary numeric values, for example, B_N 505 and the associated functions, $V_y(x)$ 510 ~~with each of these values~~ can be mutually updated and modified between a host and a remote. As will be appreciated by those skilled in the art, a number of different instructions, protocols format and methods can be employed to exchange any type of configuration parameters, consistent with the concepts and teachings of the present invention.

[0072] The packet 600 can contain any type of protocol format in it. A single bit field 550 is appended with the value of 1 to indicate that the packet 600 carries system information and not user's data. The field 555 represents the instruction type that the packet 600 carries As discussed with reference to FIGS. 9 and 10, the encryption functions 610 operates on packet 600 and the resulting packet 650 does not disclose any indication to an eavesdropper that the said packet contains user's information or system information. On the receiving side, the appropriate decryption functions are performed to restore the system information packet along with its bit value of 1 in the field 550. The receiving end processes the system packet accordingly by identifying the instruction mnemonics.

Replacement Sheet

(Amended to include the reference of the New FIG. 9A under paragraph [0026])

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The invention will be better understood with its objectives when consideration is given to the following detailed description thereof. Such a description makes reference to the annexed drawings wherein:

[0017] FIG. 1 illustrates a host which communicates to a plurality of remote devices through unsecured communication channels.

[0018] FIG. 2 presents a typical communication session between a host and remote based on the techniques presented in this invention.

[0019] FIG. 3 illustrates an example of a plurality of mathematical or logical functions defined in the first and second function pools with a unique one-to-one relationship.

[0020] FIG. 4A shows a variable size random number within a certain bit length range containing a predetermined number of bits that are located at pre-determined and specific locations within the said random number, representing Group Bits, with respect to the defined boundaries.

[0021] FIG. 4B is a table representing the numeric values of Group Bits, the corresponding number of Function Bits assigned, and their pre-determined and specific locations within the said random number.

[0022] FIG. 5A, 5B and 5C illustrate the exact position of the specific and individual Function Bits belonging to a particular Group Number from a well defined start or ending boundaries.

[0023] FIG. 6 is a table showing all the possible numeric Function Bits values along with their corresponding association with the sequence of logical and mathematical functions belonging to the first function pool.

[0024] FIG. 7 shows a table which represents the relationship between a set of binary numbers and the total number of rounds of encryption to be performed.

[0025] FIGS. 8A and 8B illustrate an example of a function operation and its inverse function operation on an information segment.

[0026] FIG. 9 and 9A depict the encryption technique using the random number and a set of logical and mathematical functions.

[0027] FIG. 10 illustrates the decryption technique utilizing the random number and the inverse logical and mathematical functions.

[0028] FIG. 11 shows exemplary frame formats that can be used to exchange configuration parameters and system information between a host and a remote processor.
[0029] FIGS. 12A & B illustrate a transparent and non-transparent means of exchanging a random number between a host and a remote processor.

[0030] FIG. 13 shows a method for diffusing the individual bits of information belonging to a password segment into an arbitrary binary bit segment.

[0031] FIG. 14 shows a table used by the mapping algorithm to diffuse the individual bits of information in a password segment.

Response to Objection 4 of the Office action:- For the clarification of "Repeating the Encryption Rounds"

The underlined Text to be inserted between paragraphs [0056] and [0057] of the original submitted specification:

FIG. 9A further illustrates the procedure as discussed in reference to FIG. 9 for repeating the rounds of encryption. As illustrated, the random number 'x' 207 is used to uniquely identified Group and Function Bits. Based on the numeric bit value as calculated from the identified Function Bits ($x_k \ldots x_2\ x_1\ x_0$) a single or plurality of functions ($f_1(x) \ldots f_n(x)$) are identified from the first pool. As mentioned, the said functions are used to operate on the said random number 'x' 207 to encrypt the said random number that further enhances its randomness characteristics. The corresponding functions ($g_1(x) \ldots g_n(x)$) 215 can be identified through a tabular relation established as illustrated in FIG. 3 or by any other . As it is evident, a person skilled in the art may choose another way to collaborate the functions being used in the first pool to encrypt random number with the functions in the second pool used to encrypt data segments. Nevertheless, any of the these techniques fall under the scope of the presented invention.

Referring back to 9A, the random number 'x' 207 encrypted by the function(s) ($f_1(x) \ldots f_n(x)$) of the first pool yields to another random number 'y' 217. Similarly, the digital data segment 205 inclusive with the single bit 210 is encrypted by ($g_1(x) \ldots g_n(x)$) 215 to yield the encrypted data segment D_g 220.

The block 204 represents a single round of encryption on the random number with a corresponding single encryption round that yields the first encrypted data segment D_g 220. Similarly, in the next encryption round as indicated by the block 206, the Group and Function Bits are identified and depending upon the calculated numeric value of the bit set ($y_g \ldots y_2\ y_1\ y_0$) the functions set ($j_1(y) \ldots j_m(y)$) from the first pool are identified . The corresponding functions in the second pool ($k_1(D_g) \ldots k_m(D_g)$) 222 are identified to perform a second round of encryption on the encrypted data segment D_g 220 from the

previous round to produce D_{gk} 224. The random number 'y' is further encrypted by the functions set $(j_1(y) \ldots j_m(y))$ to produce another random number 'w' 218. With the similar procedure, the block 223 indicates intermediate rounds of encryption being performed on the encrypted random number from the previous rounds. Similarly, the block 225 indicates the intermediate encryption rounds performed on the data segments already encrypted from the previous rounds.

The resulting random number 'z' 219 is used as the seed random number for the last encryption round as indicated by the block 208. The encrypted segment D_{ghk} 226 is produced after performing the intermediate rounds of encryption as indicated by the block 225. The functions set $(u_1(z) .. u_w(z))$ belonging to the first pool uniquely identifies the functions set belonging to the second pool to operate on the digital data segment D_{gkh} 226 as $(v_1(D_{gkh}) .. v_w(D_{gkh}))$ 235 to produce D_{gkhv} 228. Since the numbers of rounds of encryption N_T 213 are exhausted, the encryption process stops. The final encrypted digital data segment D_{gkhv} 228 is handed over to the lower communication layers which add appropriate header(s) and trailer(s) to transmit it to the host processor in the packet 300.

Response to Objection 6:- For the clarification of "First Function Pool" and its relation with "Second Function Pool".

The underlined Text to be inserted in paragraphs [0036] and [0037] of the original submitted specification

[0036] Before a communication session can utilize the encryption techniques discussed herein the host `A` 1 and the remote `B` 5 define two sets of function pools. As illustrated in FIG. 3, the first function pool 21 contains a plurality of functions, e.g., $f_1(x)$ to $u_w(z)$, which may constitute any type of mathematical or logical functions of any complexity. The term "any type of mathematical or logical functions of any complexity" means that the functions selected in the first pool can be handled and processed by the system resources utilized by the host and the remote processors. It is desired that the functions be chosen in the first pool in such a way that when the said single or plurality of functions are operated on a random number `x` they yield a higher degree of randomness in the said random number. The order or the sequence in which the functions or organized or listed in the first pool depends on the numeric value resulting from the Group and Function Bits after each round of encryption on the random number.

[0037] The second function pool 23 contains another class of plurality of functions, e.g., e.g., $g_1(x)$ to $v_w(z)$, with the criteria such that there exists a unique inverse function for each of the functions defined in the second pool. FIG. 3 shows a table illustrating the functions defined in the second pool. As illustrated, functions defined in the first pool 21 do not have any correlation with the functions 25 defined in the second pool 23. In other words, the sequence of functions that is assembled to populate the first pool can be the same or completely different from the sequence of functions 25 in the second pool 23. In a preferred embodiment, it is recommended that the defined functions 25 in the second pool 23 should be independently chosen and tabulated, and should not have any dependency on the order of the functions chosen in the first pool 21. Every function 25 defined in the second pool 23 has one to one mapping and unique correspondence to its counterpart inverse function 27. For example, if a function 'g_1' operates on a digital

information segment of an arbitrary length 'D_x', it changes the contents of the said segment yielding another segment 'D_y'. If its counterpart inverse function 'g^{-1}' operates on D_y, by its definition, it will restore the original digital information segment D_x. FIG. 3 illustrates a range of function defined (g_1 to v_w) in the second pool with the condition that every function defined there must contain its inverse function, i.e., g^{-1} to v_w^{-1}. By definition, the result of a function can be restored to its original value through its inverse function. Mathematically speaking, a function H(x)=y has its inverse function (H.sup.-1) if H.sup.-1(y)=x. The common examples of reversible functions are addition, with its inverse as subtraction; right rotation function with its inverse as left rotation function, etc. The solutions to linear and polynomial equations can also be categorized as reversible functions only if a predetermined unique solution (single root of the equation) is chosen at both sides. The functions defined in the second pool can also be consist of any type of mathematical or logical functions of any complexity with the condition that there must exist an inverse function for every function defined in the said pool. Each of the function defined in the first pool uniquely correspond to a function defined in the second pool. In other words, the order in which these functions need to be placed in the first pool can be determined through the numeric outcomes of the Group and Function Bits after each round of encryption. The numeric outcomes of the Group and Function Bits resulting after each round of encryption can be used to determine the order in which the functions in the second pool are laid out. It is recommended to define another set of Group and Function Bits that exclusively determine the order in which the functions 25 defined in the second pool 23 are laid out. As mentioned earlier that the order in which the functions in the first pool 21 are laid out can be the same or entirely different in which the functions 25 in the second pool 23 are laid out.